P4 Expert Roundtable Series
April 28-29, 2020
Hosted by: ONF

Offloading Media Traffic to P4 Programmable Data Plane Switches

Elie Kfoury[1] **(Ph.D. Student)**, Jorge Crichigno[1], Elias Bou-Harb[2]

[1] University of South Carolina, Columbia, SC

[2] University of Texas, San Antonio, TX

Supported by the NSF awards 1925484 and 1829698

Special thanks to **Vladimir Gurevich** (Barefoot Networks, an Intel Company) for his insightful feedback on various technical issues.

Agenda

- Introduction
- Background Information
  - Session Initiation Protocol (SIP) and Real Time Protocol (RTP)
  - Network Address Translation (NAT) traversal problem
  - P4 switches
- Proposed solution
- Evaluation
- Lessons learned

Introduction

- According to estimations, media traffic represents approximately 80% of the total traffic over the Internet[1]
- Much media traffic is generated by end users communicating with each other
- Media services (voice, video) running alongside the data network in campuses are becoming standard

---

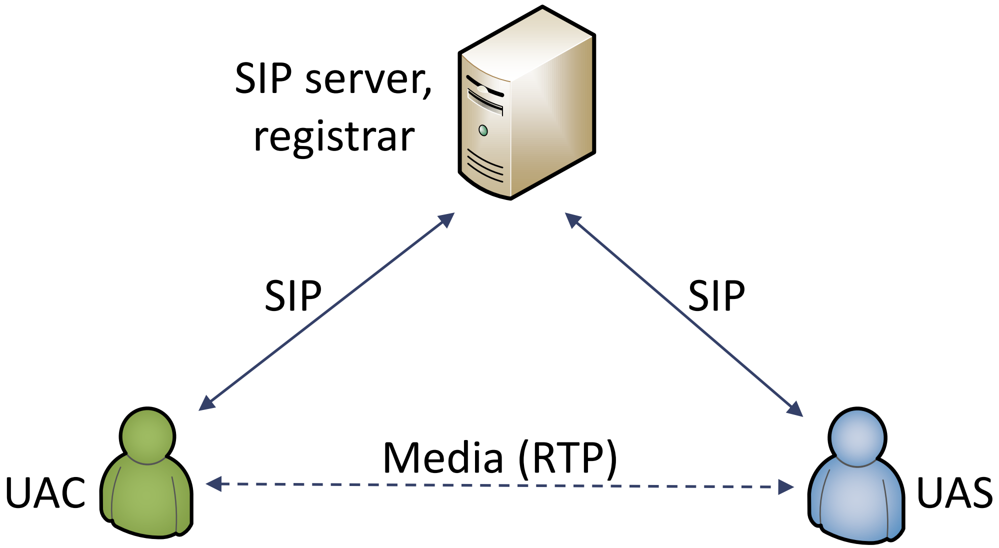
[1] H. W. Barz and G. A. Bassett, Multimedia networks: protocols, design and applications, John Wiley and Sons, 2016.

Voice and Video

- Conversational Voice- and Video-over-IP are widely used today
  - Open and proprietary solutions
- Supporting protocols are divided into two main categories
  - Session control protocols (signaling): establish and manage the session
  - E.g., Session Initiation Protocol (SIP)
- Media protocols (media)
  - Transfer audio and video streams between the end-users
  - E.g., Real Time Protocol (RTP)
- Desirable Quality-of-Service (QoS) characteristics
  - Delay- and jitter-sensitive, low values
  - Occasional losses are tolerated

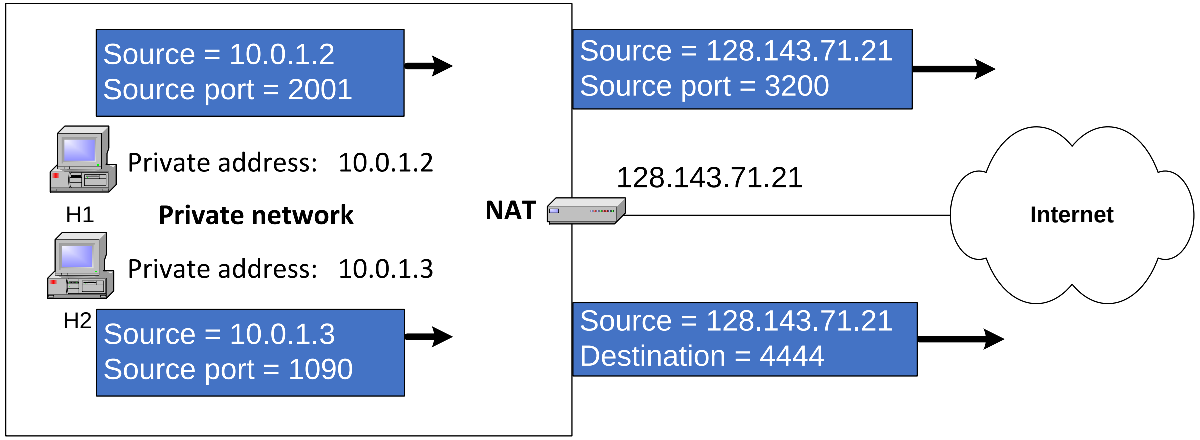
Signaling and Media Protocols

- SIP initiates, maintains, and terminates multimedia sessions between endpoints
  - User agent client (UAC)
  - User agent server (UAC)
- RTP transports real-time data, such as audio and video



Network Address Translation (NAT)

- NAT maps ports, private IP addresses to public IP addresses
  - Used in campus / enterprise networks, operators[1]
- NAT introduces various issues
  - Violation of the end-to-end principle
  - Traversal of end-to-end sessions



[1] I. Livadariu et al., “Inferring carrier-grade NAT deployment in the wild,” in IEEE 2018 INFOCOM, 2018.

Network Address Translation (NAT)

- NAT prevents a user from outside from initiating a session
- If both users have NATs, then neither can accept a call
  - IP translation is recorded by a SIP registrar server
- SIP carries the IP addresses and ports to be used by RTP to send/receive media
- Ports are unknown until RTP traffic starts
- Several solutions proposed for NAT traversal
  - STUN - RFC 5389[1], TURN - RFC 7566[2], ICE - RFC 8445[3]

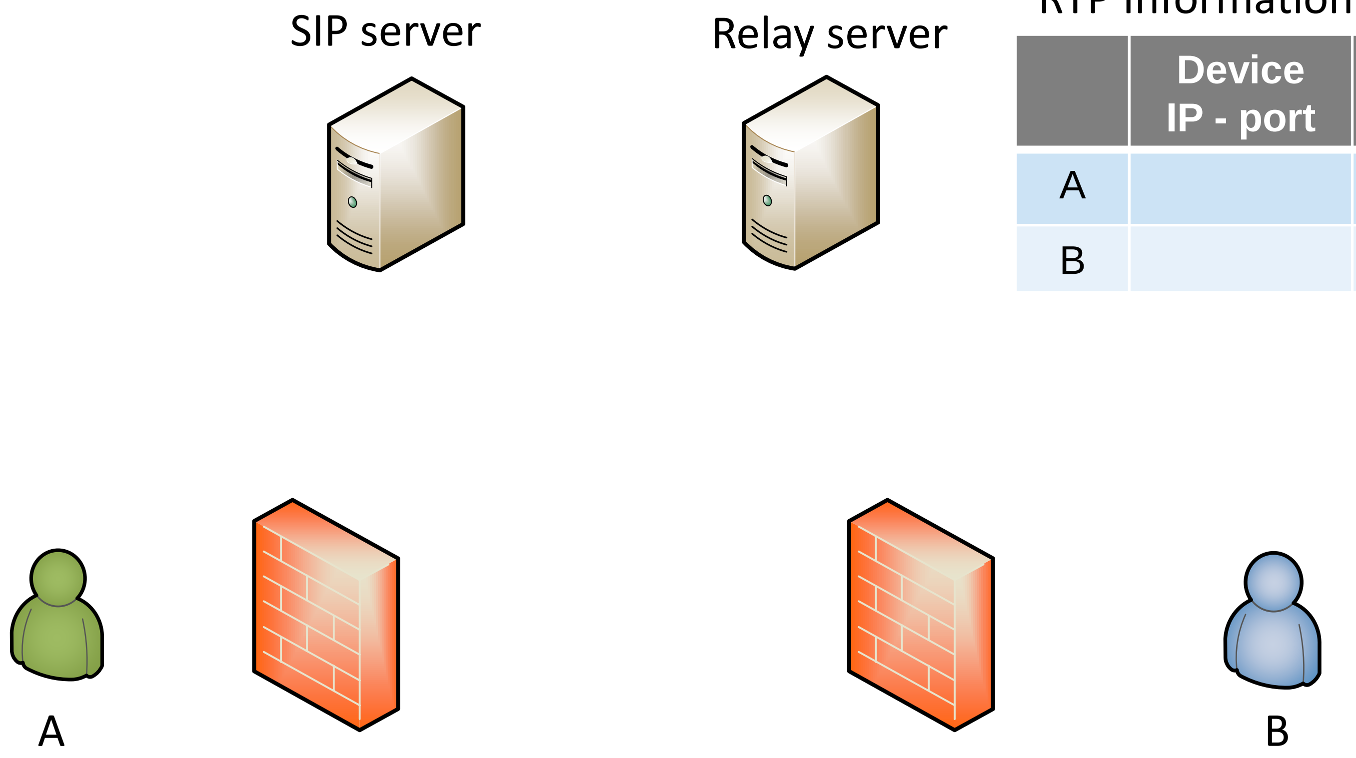
[1] D. Wing, P. Matthews, R. Mahy, and J. Rosenberg, "RFC 5389 - STUN: Session traversal utilities for NAT," 2008.

[2] M. Petit-Huguenin, S. Nandakumar, G. Salgueiro, and P. Jones, "RFC 7566 - TURN: Traversal using relays around NAT (TURN) uniform resource identifiers," 2013.

[3] J. Rosenberg and C. Holmberg, "RFC 8445 - ICE: Interactive connectivity establishment: a protocol for Network Address Translator (NAT) traversal," 2018.

Relay Server for Media Traffic

- Intermediary device

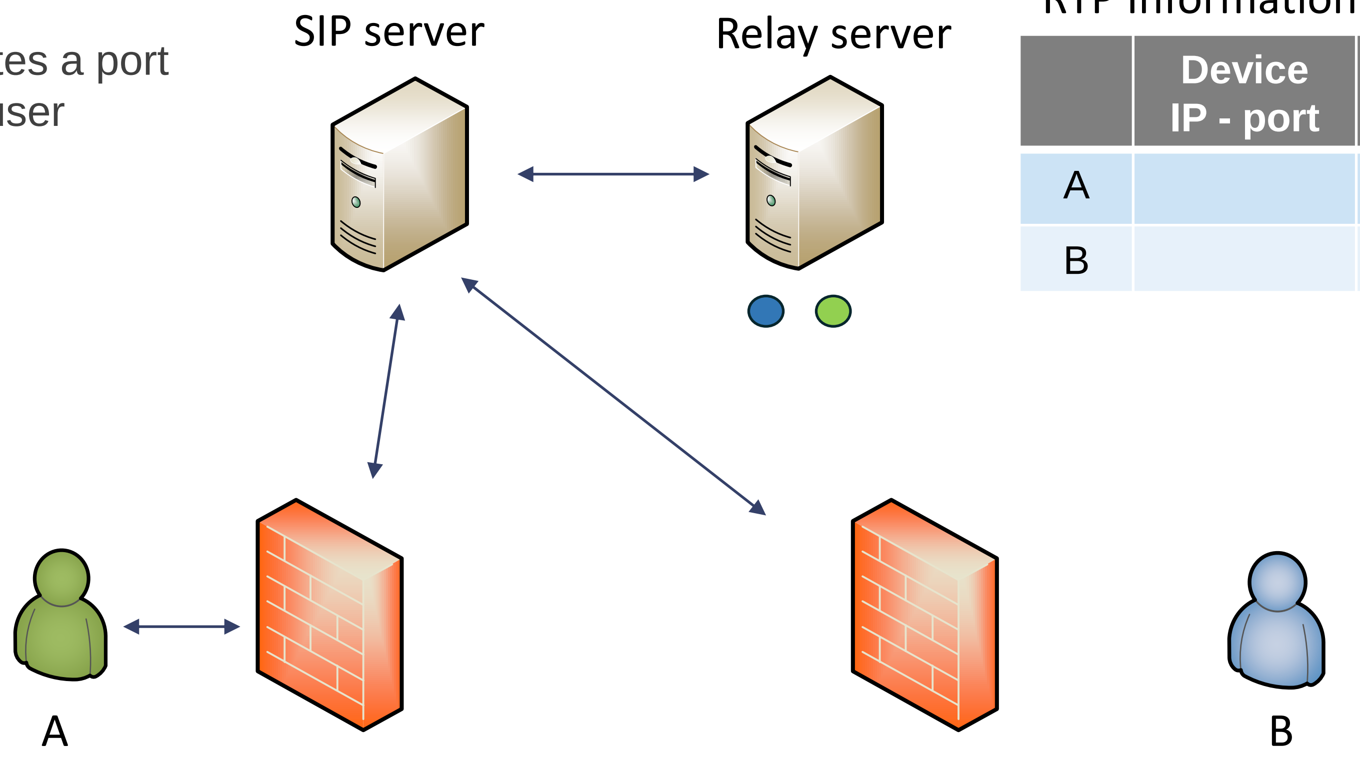


RTP Information at relay server

	Device IP - port	Allocated IP - port
A		
B		

Relay Server for Media Traffic

- Intermediary device
- SIP establishes the session
  - RTP ports are unknown
  - The relay server allocates a port on behalf of each end user

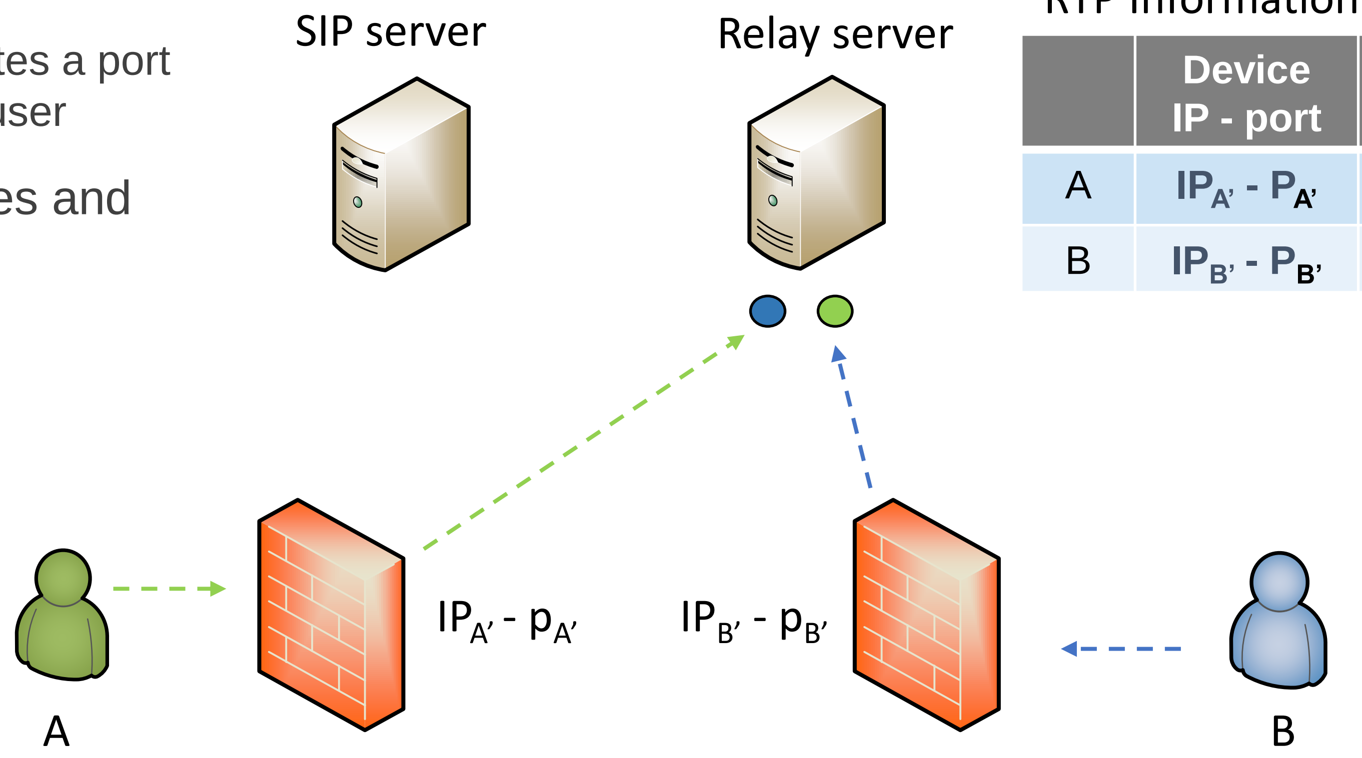


RTP Information at relay server

	Device IP - port	Allocated IP - port
A		IP_R - P_{RA}
B		IP_R - P_{RB}

Relay Server for Media Traffic

- Intermediary device
- SIP establishes the session
  - RTP ports are unknown
  - The relay server allocates a port on behalf of each end user
- The relay server receives and relays the RTP traffic

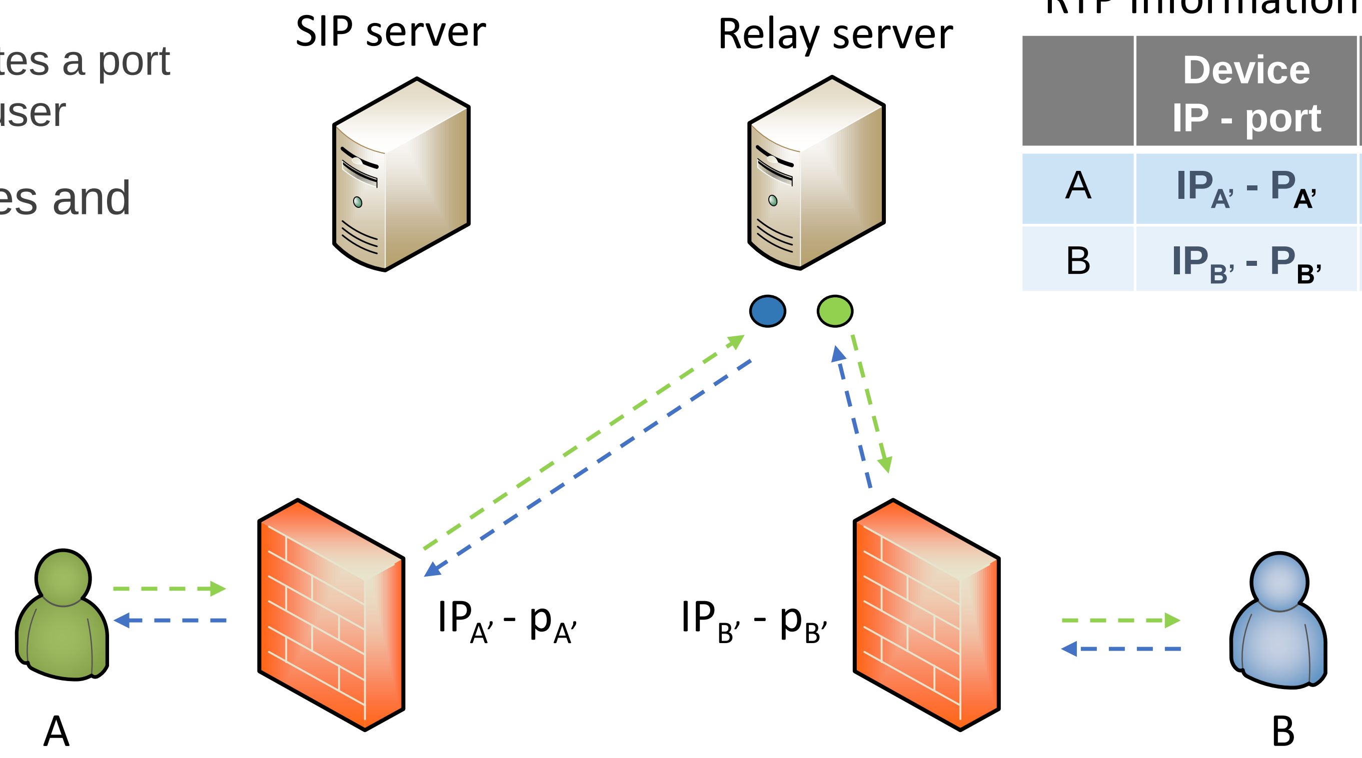


RTP Information at relay server

	Device IP - port	Allocated IP - port
A	$IP_{A'}$ - $P_{A'}$	IP_R - P_{RA}
B	$IP_{B'}$ - $P_{B'}$	IP_R - P_{RB}

Relay Server for Media Traffic

- Intermediary device
- SIP establishes the session
  - RTP ports are unknown
  - The relay server allocates a port on behalf of each end user
- The relay server receives and relays the RTP traffic



RTP Information at relay server

	Device IP - port	Allocated IP - port
A	$IP_{A'}$ - $P_{A'}$	IP_R - P_{RA}
B	$IP_{B'}$ - $P_{B'}$	IP_R - P_{RB}

Overview P4 Switches

- P4 switches permit programmer to program the data plane
- Add proprietary features; e.g., emulate RTP relay server
  - Parse packet headers, including UDP packets carrying RTP traffic
  - Header inspection, identifying media sessions using the 5-tuple
  - Modify fields, IP addresses and ports
- If the P4 program compiles, it runs on the chip at line rate

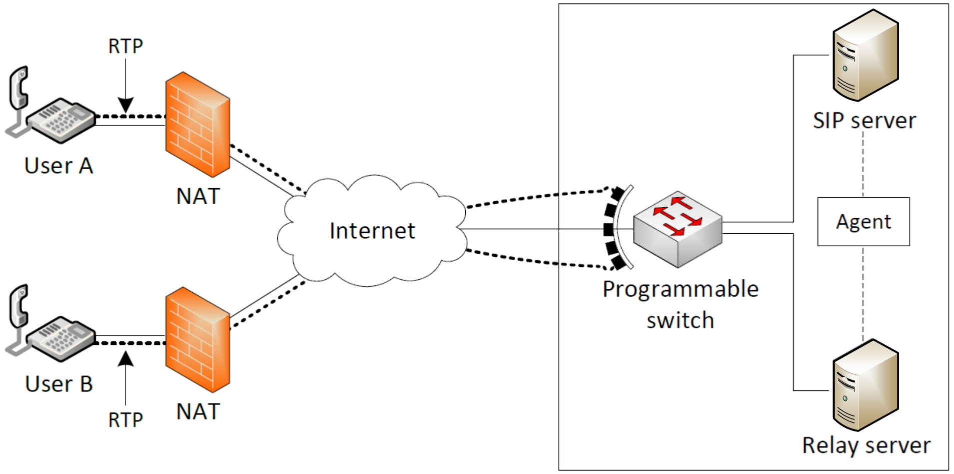
```
/*************************************************************************
*********************** P A R S E R  ***********************************
*************************************************************************/

    state parse_ethernet {
        packet.extract(hdr.ethernet);
        transition select(hdr.ethernet.etherType) {
            TYPE_IPV4: parse_ipv4;
            default: accept;
        }
    }

    state parse_ipv4 {
        packet.extract(hdr.ipv4);
        verify(hdr.ipv4.ihl >= 5, error.IPHeaderTooShort);
        transition select(hdr.ipv4.ihl) {
            5           : accept;
            default     : parse_ipv4_option;
        }
    }
```

P4 code

Programmable chip

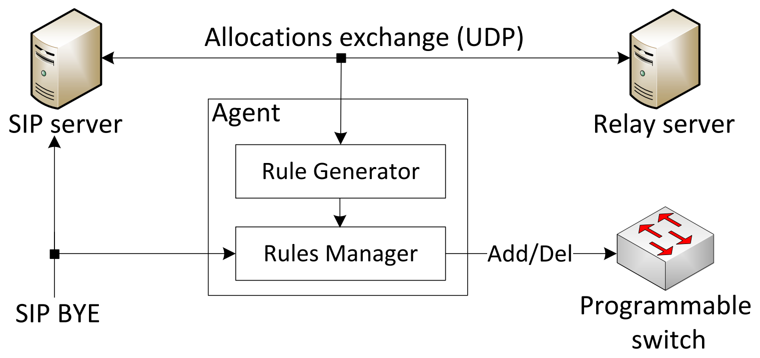
Proposed System

- Emulate the behavior of the relay server using programmable switch:
  1. Parse the incoming packet carrying media traffic from the first party, say user A
  2. Identify the session this packet belongs to by using the 5-tuple
  3. Replace the source IP with that of the relay server, and the source port with that used by the relay server to receive traffic from user A
  4. Replace the destination IP and the destination port with those of user B
  5. Recalculate both IPv4 and UDP checksums
  6. Forward the packet to user B



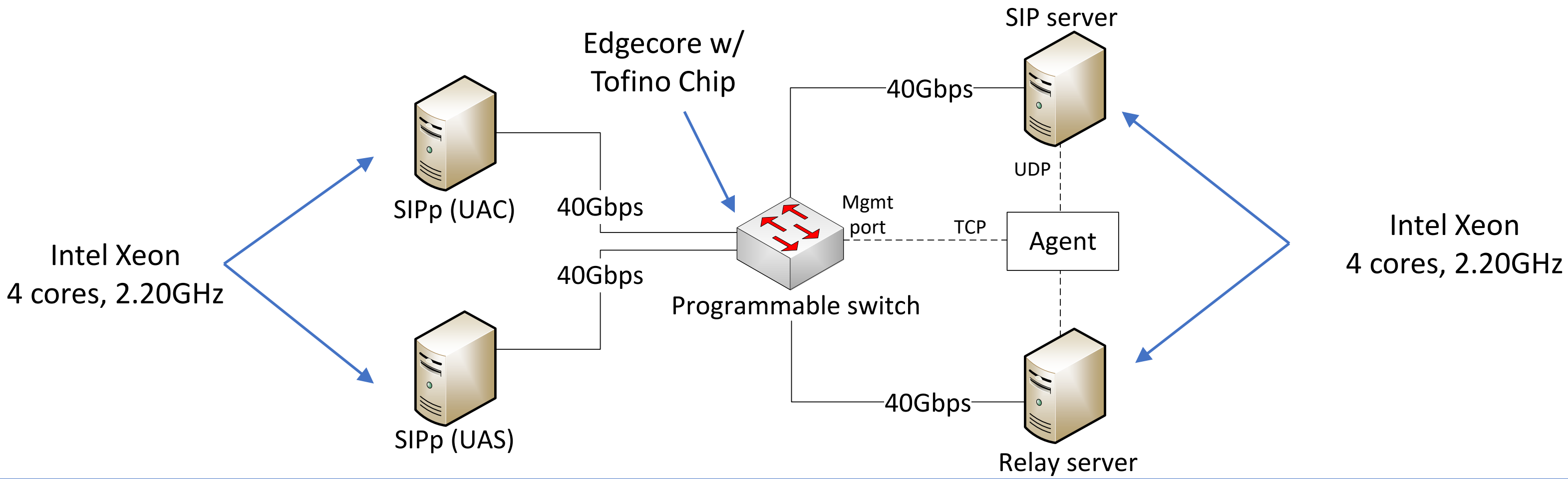
Proposed System

- A custom software (agent) learns the ports allocated to a media session by the relay server
- The Rule Generator uses the 5-tuple allocated to the media session to construct a unique session identifier
- It stores identifiers of the media sessions and the new header' values in the switch
- It also clears media sessions allocated in the switch when a call is teared down



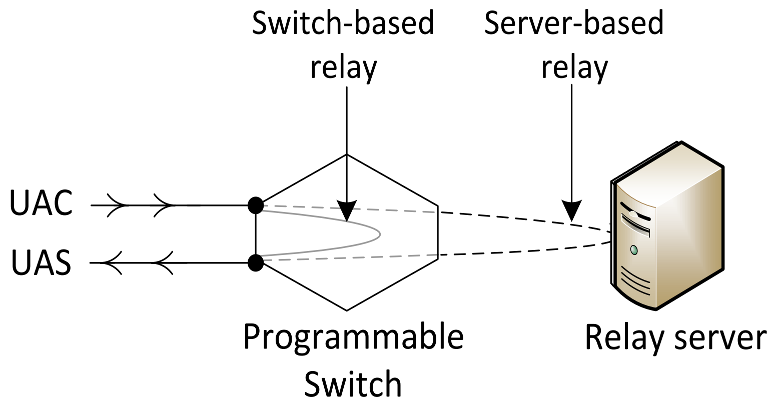
Implementation and Evaluation

- OpenSIPS, an open source implementation of a SIP server
- RTPProxy, a high-performance relay server for RTP streams
- SIPp: an open source SIP traffic generator that can establish multiple concurrent sessions and generate media (RTP) traffic
- Iperf3: traffic generator used to generate background UDP traffic
- Edgecore Wedge100BF-32X: programmable switch



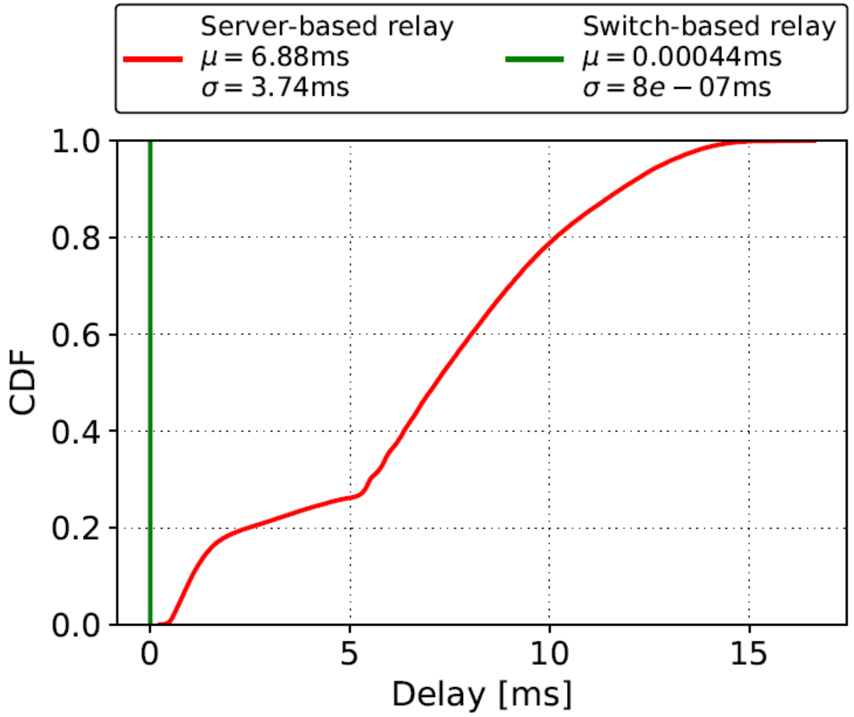
Implementation and Evaluation

- Two scenarios are considered:
  - “Server-based relay”: relay server is used to relay media between end devices, without the intervention of the switch
  - “Switch-based relay”: the switch is used to relay media
- UAC (SIPp) generates 900 media sessions, 30 per second
- The test lasts for 300 seconds
- G.711 media encoding codec (160 bytes every 20ms)



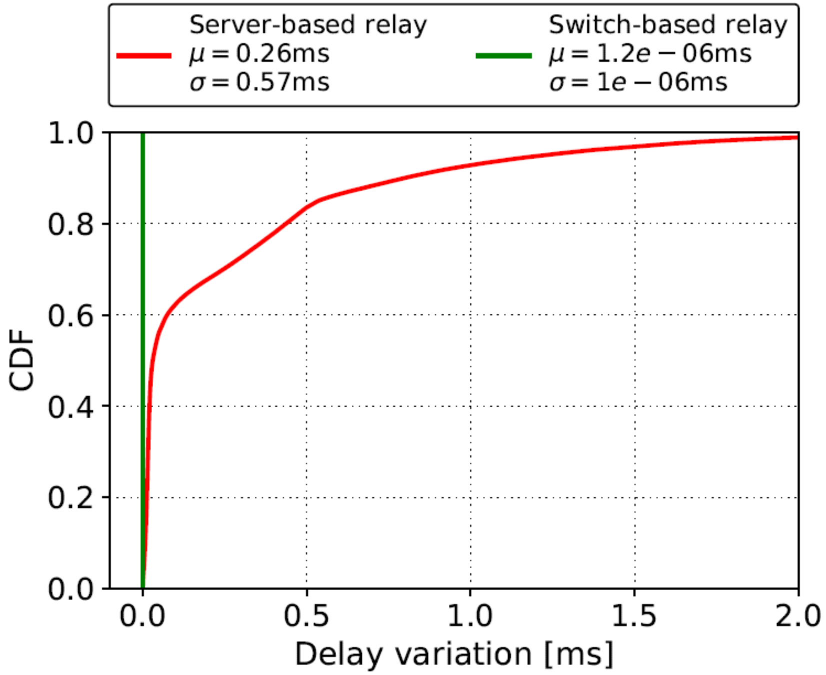
Results

- Delay: time interval starting when a packet is received from the UAC by the switch's ingress port and ending when the packet is forwarded by the switch's egress port to the UAS
  - Delay contributions of the switch and the relay server



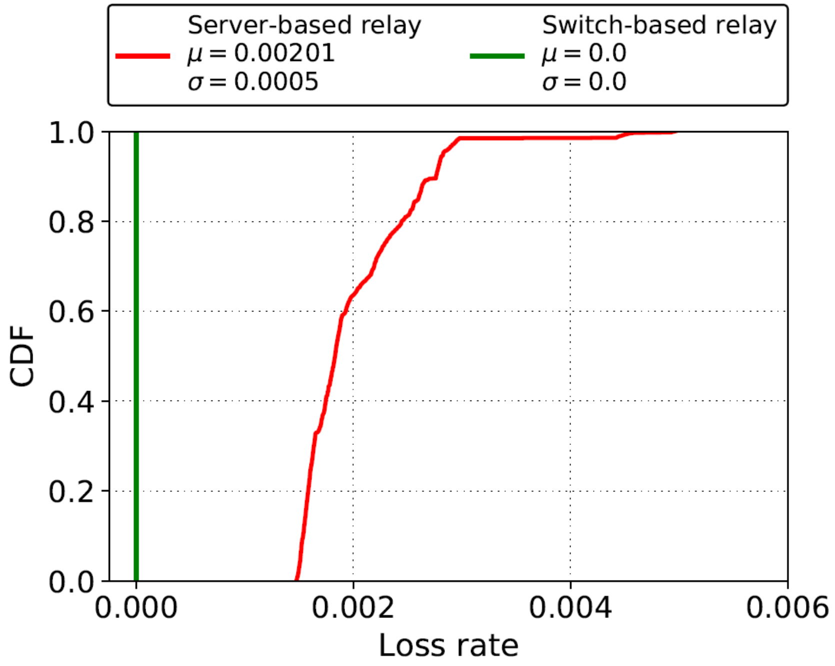
Results

- Delay variation: the absolute value of the difference between the delay of two consecutive packets
  - Analogous to jitter, as defined by RFC 4689



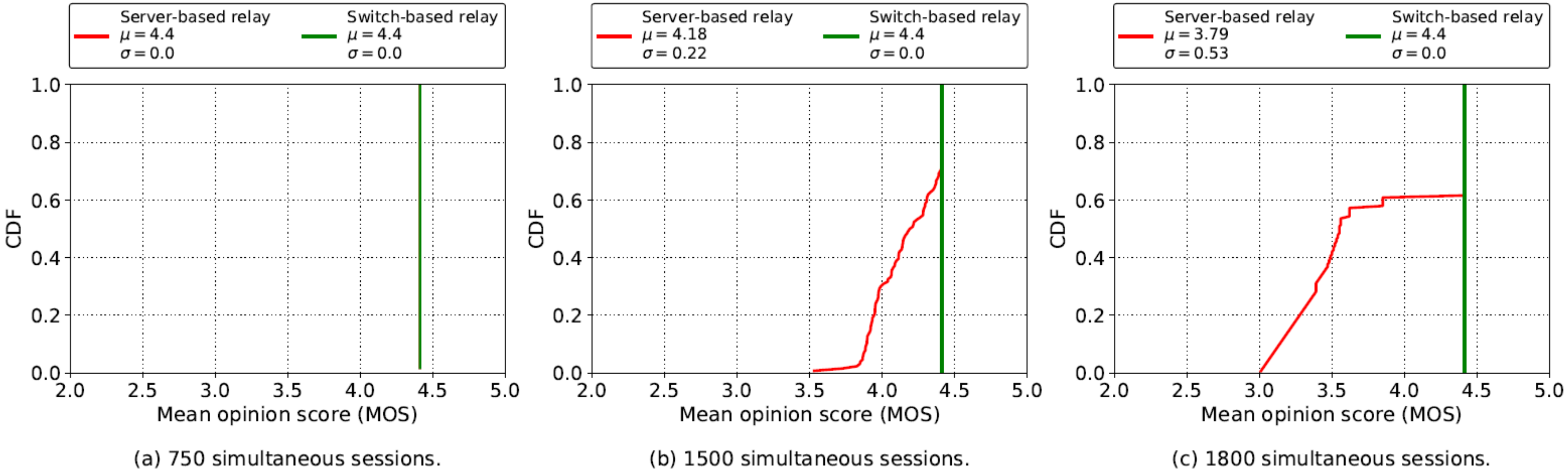
Results

- Loss rate: number of packets that fail to reach the destination
  - Calculation is based on the sequence number of the RTP header



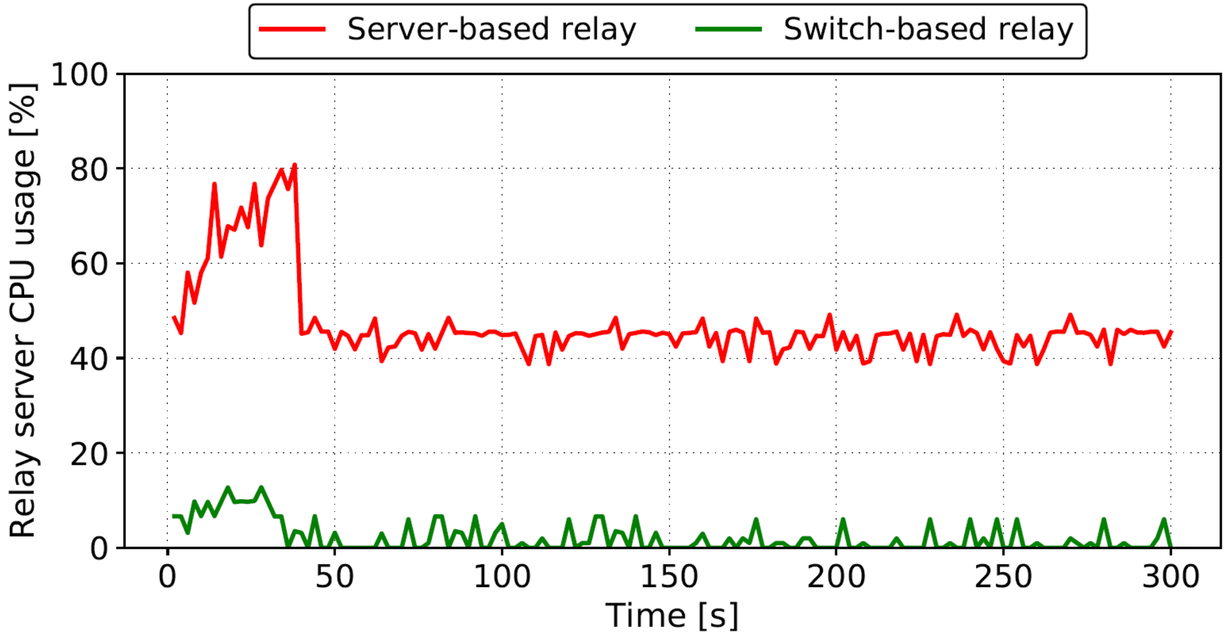
Results

- Mean Opinion Score (MOS): estimation of the quality of the media session
  - A reference quality indicator standardized by ITU-T
  - Maximum for G.711 is ~4.4



Results

- CPU usage: the percentage of the CPU's capacity used by the relay server



Resource Consumption

- The prototype is implemented in two different scenarios:
  - On top of the baseline switch program (switch.p4): implements various features including Layer 2/3 functionalities, ACL, QoS, etc.
  - Standalone implementation

On top of switch.p4			
Table size	**SRAM**	**Hash Bits**	**TCAM**
32,000	+8.45%	+2.7%	+0%
64,000	+16.2%	+4.6%	+0%

Standalone program			
Table size	**SRAM**	**Hash Bits**	**TCAM**
500,000	———-	———-	———-
1,000,000	+97.84%	+86.4%	+0%
1,050,000	+107.5%	+89.8%	+0%

Additional hardware resources used when the solution is deployed on top of switch.p4 and as a standalone program

Lessons Learned

- Advantages of using a switch-based relay:
  - Performance: ~1,000,000 sessions vs ~1,000 sessions per core
  - Optimal QoS parameters: delay, delay variation, packet loss rate
  - Flexibility: switch permits to modify / forward packets using non-standard fields
  - Precise timing information: measuring delay and its variation on the P4 switch results in precise high-resolution timing information
  - Programmer can free unused resources and customize program: accommodate additional sessions
- Limited resources
- Avoid complex application logic

Acknowledgement

- Thanks to the National Science Foundation (NSF)!
- Activities in the CI Lab at the UofSC are supported by NSF, Office of Advanced Cyberinfrastructure (OAC), awards 1925484 and 1829698

Thank You

Contact info for further questions

ekfoury@email.sc.edu

jcrichigno@cec.sc.edu

Full text

https://tinyurl.com/wab7yej

CI Lab website

http://ce.sc.edu/cyberinfra/